7. Типове за знак и низ

Проф. д-р Емил Хаджиколев

1. Типове и литерали за знаци и низове
2. Ескейп-последователности
3. Основни оператори при работа със знаци…
4. …и низове

Тип за знак в C++

- По подразбиране с тип char се представят символи от ASCII кодовата таблица.
- Символите се използват и като елементи на низове (от тип string).
- С функцията **setlocale(<константа_за_категория>, <констата_за_локализация>) от библиотеката clocale,** стартирана в началото на програмата може да се описват и символи и низове от други ЕЕ в UNICODE формат; По следния начин може да локализираме (зададем език за използваните текстове в) програмата за български език:

```
setlocale(LC_ALL, "Bulgarian"); или
setlocale(LC_ALL, "bg");  или други варианти
```

Тип за знак в C++ - локализация

- категории: http://en.cppreference.com/w/cpp/locale/LC_categories
- Примерни локализации: en, en_US, en_GB, bg, bg_BG, en_US.UTF-8, de_DE.
- В низовете за локализация участват:
  - дву-буквени ISO кодове за език (или съответните им пълни наименования) - http://www.loc.gov/standards/iso639-2/php/code_list.php;
  - дву-буквени ISO кодове за държава - https://en.wikipedia.org/wiki/ISO_3166-1;
  - кодировка;
  - и др.
  - още за кодовете: https://msdn.microsoft.com/en-us/library/ee825488(v=cs.20).aspx;
- Обектите за локализация няма да разглеждаме подробно. Те се описват като низ, но са много по-сложни неща и съдържат множество различни характеристики, които варират (при различните обекти): език; държава; регион; кодировка; представяне на числа, валута, дати и др.
- библиотеката clocale не е нужно да се включва изрично в програмата, ако тя се ползва в други включени библиотеки;

Тип за знак в C++ - литерали

Символ може да се представи по различни начини като **литерал**, заграден в апострофи:

- ***единичен символ*** – 'a', 'b' (може и български при включена локализация);
- с ***код на символ*** – '\u0061', '\u0062' ('a' и 'b' в UNICODE) – кода се задава с ляво наклонена черта '\', последвана от u и четири шестнайсетични цифри;
- ***ескейп (escape) последователност*** – '\специален символ' – предоставят алтернативен запис на специални символи, които не винаги могат да се запишат като обикновен символ.
- и др.

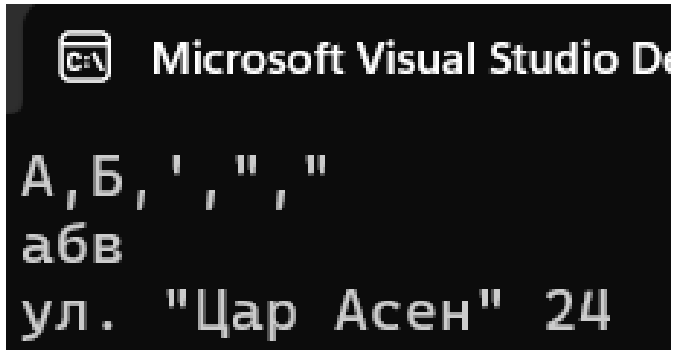
Някои ескейп последователности

ескейп последователност	символ
\t	табулация
\n	преминаване на нов ред
\'	апостроф
\"	кавичка
\\	ляво наклонена черта
\0	NULL

Тъй като апострофите, кавичките и ляво наклонената черта са специални символи, в определени случаи е необходимо да се ескейпнат.

Пример за символи и низове

```
#include <iostream>
#include <clocale>
using namespace std;
int main() {
    setlocale(LC_ALL, "bg"); // локализация за български език

    char буквaA = 'A';       // българска буква А
    char буквaБ = 'Б';       // българска буква Б
    char апостроф = '\'';
    char кавичка1 = '\"';    // кавичка като символ...
    char кавичка2 = '"';     // ... може да не се ескейпва

    cout << буквaA << ',' << буквaБ << ',' << апостроф << ',' << кавичка1 << ',' << кавичка2 << '\n';

    cout << "\u0430\u0431\u0432" << '\n';     // абв

    cout << "ул. \"Цар Асен\" 24" << "\n";  // В низ, кавичките трябва се ескейпват

    return 0;
} // '\n' е символ за край на ред, а "\n" – низ, в който има символ за край на ред
  // за низ от един символ се заделя повече памет, отколкото за един символ
```



Низ

- Низът е последователност от символи;
- В повечето ЕП (и C++) литералите за низ се ограждат в кавички;
- Типът за низ в C++ е string и е дефиниран в библиотека със същото име. string е клас (сложен тип), но тъй като е често използван, в ЕП се създават улеснения за работа с величини от този тип (за да се работи по-лесно с него). Поради това работата с величини от тип string, прилича на работа с другите примитивни (скаларни) типове.

Задаване на стойност на променлива от тип string

```
string firstName = ""; // инициализиране с празен низ
string secondName;     // автоматична инициализация с празен низ
// т.е. за разлика от примитивните типове, низовете автоматично се инициализират

string lastName = "Иванов";// неявно извикване на т.нар. конструктор (вид метод на клас)
string lastName2("Петрова");            // неявно извикване на конструктор
string lastName3 = string("Димитрова"); // явно извикване на конструктор

firstName = "Андрей";  // задаване на нови стойности
secondName = "Петров";
```

Масив от символи – низове, завършващи с NULL

- На ниско ниво литералът за низ се представя като масив от символи (за масиви ще говорим в следваща лекция по-подробно).
- Може да създадем променлива за такъв низ по следния примерен начин:

```
char name[10] = "Станислав";
```

  - name е името на променливата;
  - след името задаваме в квадратни скоби максималният брой символи, които очакваме да се записват в низа;
  - в максималния брой се предвижда място за символа за край '\0', с който се идентифицира края на низа.
  - само при декларацията, с оператор „=“ може да се задава стойност на низа, като последващо задаване на нова стойност се извършва чрез функция (strcpy, но не и с „=“).
  - ако символите на инициализиращия литерал са по-малко от максималния брой, в низа остават свободни елементи, ако са повече - може да възникнат проблеми при изпълнение на програмата (при по старите версии на компилаторите);

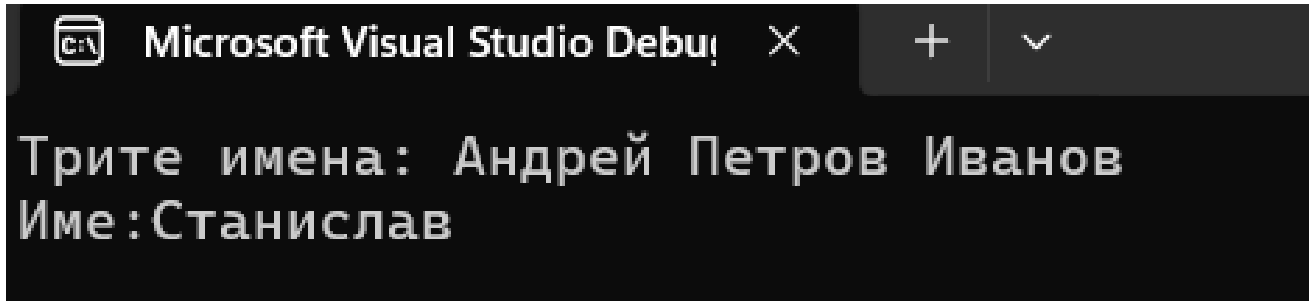
char[] и string

- Работата с низове от тип string е по-лесна, а с char[] – по-трудна:
  - при char[] има ограничения в размера, които се задават като константа (напр., char[10]), а при string може динамично да се променя броят на символите;
  - Работата с низове от тип char[] става чрез функции, а със string се работи почти както с променливи от примитивен тип.

- Някои ЕП (Java) използват като базова кодировка UTF-8 (имат късмета да са създадени по-късно) и нямат усложнения с използването на символи от различни ЕЕ. Типът char при тях е с дължина 2 байта и представят директно UNICODE символи.

char[] и string - пример

```
#include <iostream>
#include <string>

using namespace std;

int main(){
	setlocale(LC_ALL, "bg"); // стандартна функция за задаване на локализация на български език

	string firstName = "Андрей";
	string secondName = "Петров";
	string lastName = "Иванов";
	char name[10] = "Станислав";

	string fullName = firstName + " " + secondName + " " + lastName; // конкатенация с оператор +

	cout << "Трите имена: " + fullName + '\n';  // конкатенация с оператор +

	cout << "Име:" << name << '\n';          // name не може да се конкатенира с +

	return 0;
}
```



Още за типа char (1)

- Един ASCII символ от тип char се записва в 1 байт (8 бита);
- Базовите ASCII символи са с номера от 0 до 127, т.е. за представянето им се използват само 7 бита (2^7=128).
- ASCII символите може да се представят в апострофи или да се задават чрез съответните им цифрови кодове:

```
char c1 = 'b';
char c2 = 100; // ASCII кода на 'd' е 100

cout << c1 << " - " << c2 << '\n';
// отпечатва се b - d

cout << (int)c1 << " - " << (int)c2 << '\n';
// отпечатват се кодовете на символите 98 - 100
```

Още за типа char (2)

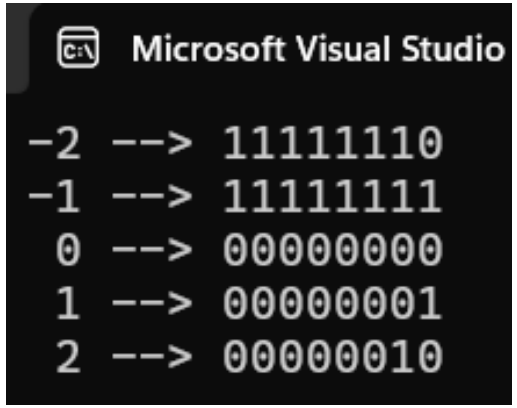
- Възможността за записване на символи на различни езици (в UNICODE) е добавена във версия C++11;
- При това всеки символ се представя чрез няколко байта – като масив char[] (многоезичният низ е масив от масиви). Добавени са и други типове за работа с UTF-16 и UTF-32 символи:
  - с char[] се представят UTF-8 символи (променлив брой байтове);
  - char16_t[] – UTF-16 (2 байта)
  - char32_t[] – UTF-32 (4 байта)

Използване на char за работа с числа

- В променливи от тип char може да записваме числа (както видяхме).
- дефиниционната област на типа char е [-128, 127];
- т.е. за променливи от тип char може да задаваме числа със знак + или -;
- в паметта, знакът при целочислените числа се определя (обикновено) от първия (старши) бит:
  - 0 – положително число;
  - 1 – отрицателно число;

Побитово представяне на числа - пример

```cpp
#include <iostream>
#include <bitset>

using namespace std;

int main()
{
    cout << "-2 --> " << bitset<8>(-2) << '\n';
    cout << "-1 --> " << bitset<8>(-1) << '\n';
    cout << " 0 --> " << bitset<8>(0) << '\n';
    cout << " 1 --> " << bitset<8>(1) << '\n';
    cout << " 2 --> " << bitset<8>(2) << '\n';

    return 0;
}
```



Представяне на числа с char

С помощта на ключови думи на езика C++ се указва дали числовите типове са със знак или без знак

- [signed] char:
  - 8 битови числа със знак в интервал [-128, 127] = [-2^7, 2^7-1];
  - signed се задава по подразбиране;
  - първият бит определя знака (1-минус, 0-плюс);
- unsigned char:
  - 8 битови числа без знак в интервал [0, 255]=[0, 2^8-1];
  - няма бит за знак.

Някои операции за символи

- Символите имат числови представяния в кодовите таблици.
- Буквите от ЕЕ в кодовите таблици са подредени лексикографски – по азбучен ред – „АБВГДЕ...“, после малки букви „абвгде…“
- Може да се извършват различни **действия** върху символите **чрез съответните им числови представяния** (в кодовите таблици);
  - ***сравнения*** – с оператори за сравнение (<, >, <=, >=, ==, !=), които ще разгледаме в следваща лекция;
  - ***аритметични операции*** – само операторът –(минус) (и + донякъде) имат смисъл – с минус може да се определи близостта на символите (в кодовата таблица);

Операции върху низове (1)

- Тъй като с низовете от тип char[] се работи по-трудно, ние ще работим основно с тип string.
- В езика C се използват само низове от тип char[], тъй като няма обектно-ориентирани възможности.
- **Някои основни операции** върху низове от тип string са:
  - ***сравнения*** – с оператори за сравнение, сравнението се извършва лексикографски: сравняват се символ по символ (от ляво надясно), докато се уточни резултата от сравнението

```
cout << ("abc" < "abd") << '\n';   // 1 – при третите символи се разбира резултата
cout << ("abc" > "abd") << '\n';   // 0 - при третите символи се разбира резултата
cout << ("abc" < "abcde") << '\n'; // 1 - при четвъртите символи се разбира резултата
cout << ("abc" < "bcd") << '\n';   // 1 - при първите символи се разбира резултата
```

Операции върху низове (2)

- ***конкатенация (слепване на низове)*** – извършва с оператор за конкатенация - +

```
string s1 = "низ1";
string s2 = "низ2";
cout << (s1+s2) << '\n';
```

  - В резултат се образува нов низ ("низ1низ2" в примера), при който вторият низ се прилепва към края на първия;
  - Не може да се конкатенират литерали за низ

```
"низ1"+ "низ2"
```

  - Може да се конкатенира променлива от тип string с литерал за низ (защото е предефиниран оператора)

```
s1 + "низ"
"низ" + s1
```

  - Може да се конкатенира променлива от тип string с число, което е преобразувано до низ с помощта на функцията std::to_string(<числова_величина>):

```
s1 + to_string(5)
```

Работа с обект от клас string

(без разяснения)

```
#include <iostream>
#include <string>
using namespace std;

int main() {
    string name = "Иван иванов";

    setlocale(LC_ALL, "Bulgarian");

    // Дължина на низа – метод length()
    cout << "Дължина на низа: " << name.length()
<< endl;

    // Достъп до символите – първият елемент е с
индекс 0
    char firstChar = name[0];
    cout << "Първи символ: " << firstChar << endl;

    // Промяна на символ
    name[5]='И';
    cout << "Нов низ: " << name << endl;

    // Добавяне на низ
    name += " Петров";
    cout << "Нов низ: " << name << endl;

    // Извличане на подниз с метод substr
    string subString = name.substr(9, 2);
    cout << "Подниз: " << subString << endl;

    // replace не е много удобен в C++

    return 0;
}
```